regexp manual page - Tcl Built-In Commands

tcl.tk/man/tcl/TclCmd/regexp.htm

NAME

regexp — Match a regular expression against a string

SYNOPSIS

regexp ?*switches*? *exp string* ?*matchVar*? ?*subMatchVar subMatchVar ...*?

DESCRIPTION

Determines whether the regular expression *exp* matches part or all of *string* and returns 1 if it does, 0 if it does not, unless **-inline** is specified (see below). (Regular expression matching is described in the **re_syntax** reference page.)
If additional arguments are specified after *string* then they are treated as the names of variables in which to return information about which part(s) of *string* matched *exp*. *MatchVar* will be set to the range of *string* that matched all of *exp*. The first *subMatchVar* will contain the characters in *string* that matched the leftmost parenthesized subexpression within *exp*, the next *subMatchVar* will contain the characters that matched the next parenthesized subexpression to the right in *exp*, and so on.

If the initial arguments to **regexp** start with **-** then they are treated as switches. The following switches are currently supported:

-about
Instead of attempting to match the regular expression, returns a list containing information about the regular expression. The first element of the list is a subexpression count. The second element is a list of property names that describe various attributes of the regular expression. This switch is primarily intended for debugging purposes.

-expanded
Enables use of the expanded regular expression syntax where whitespace and comments are ignored. This is the same as specifying the **(?x)** embedded option (see the **re_syntax** manual page).

-indices
Changes what is stored in the *matchVar* and *subMatchVar*s. Instead of storing the matching characters from *string*, each variable will contain a list of two decimal strings

giving the indices in *string* of the first and last characters in the matching range of characters.

-line
Enables newline-sensitive matching. By default, newline is a completely ordinary character with no special meaning. With this flag, "[^" bracket expressions and "." never match newline, "^" matches an empty string after any newline in addition to its normal function, and "$" matches an empty string before any newline in addition to its normal function. This flag is equivalent to specifying both **-linestop** and **-lineanchor**, or the **(?n)** embedded option (see the **re_syntax** manual page).

-linestop
Changes the behavior of "[^" bracket expressions and "." so that they stop at newlines. This is the same as specifying the **(?p)** embedded option (see the **re_syntax** manual page).

-lineanchor
Changes the behavior of "^" and "$" (the "anchors") so they match the beginning and end of a line respectively. This is the same as specifying the **(?w)** embedded option (see the **re_syntax** manual page).

-nocase
Causes upper-case characters in *string* to be treated as lower case during the matching process.

-all
Causes the regular expression to be matched as many times as possible in the string, returning the total number of matches found. If this is specified with match variables, they will contain information for the last match only.

-inline
Causes the command to return, as a list, the data that would otherwise be placed in match variables. When using **-inline**, match variables may not be specified. If used with **-all**, the list will be concatenated at each iteration, such that a flat list is always returned. For each match iteration, the command will append the overall match data, plus one element for each subexpression in the regular expression. Examples are:

```
regexp -inline -- {\w(\w)} " inlined "
      → in n
regexp -all -inline -- {\w(\w)} " inlined "
      → in n li i ne e
```

-start *index*
Specifies a character index offset into the string to start matching the regular expression at. The *index* value is interpreted in the same manner as the *index* argument to **string index**. When using this switch, "^" will not match the beginning of the line, and \A will still match the start of the string at *index*. If **-indices** is specified, the indices will be indexed starting from the absolute beginning of the input string. *index* will be constrained to the bounds of the input string.

--
Marks the end of switches. The argument following this one will be treated as *exp* even if it starts with a **-**.

If there are more *subMatchVar*s than parenthesized subexpressions within *exp*, or if a particular subexpression in *exp* does not match the string (e.g. because it was in a portion of the expression that was not matched), then the corresponding *subMatchVar* will be set to "**-1 -1**" if **-indices** has been specified or to an empty string otherwise.

EXAMPLES

Find the first occurrence of a word starting with **foo** in a string that is not actually an instance of **foobar**, and get the letters following it up to the end of the word into a variable:

```
regexp {\mfoo(?!bar\M)(\w*)} $string -> restOfWord
```

Note that the whole matched substring has been placed in the variable "**->**", which is a name chosen to look nice given that we are not actually interested in its contents.

Find the index of the word **badger** (in any case) within a string and store that in the variable **location**:

```
regexp -indices {(?i)\mbadger\M} $string location
```

This could also be written as a *basic* regular expression (as opposed to using the default syntax of *advanced* regular expressions) match by prefixing the expression with a suitable flag:

```
regexp -indices {(?ib)\<badger\>} $string location
```

This counts the number of octal digits in a string:

```
regexp -all {[0-7]} $string
```

This lists all words (consisting of all sequences of non-whitespace characters) in a string, and is useful as a more powerful version of the **split** command:

```
regexp -all -inline {\S+} $string
```